<http://www.devrecipes.com/2010/02/14/datetimetimezone-manipulation-and-conversion-in-java-a-few-examples/>

Date/Time/Timezone manipulation and conversion in Java: a few examples

February 14th, 2010 in [Java/J2EE](http://www.devrecipes.com/category/java-j2ee/). [6 comments](http://www.devrecipes.com/2010/02/14/datetimetimezone-manipulation-and-conversion-in-java-a-few-examples/#respond)

7 votes, average: 4.57 out of 57 votes, average: 4.57 out of 57 votes, average: 4.57 out of 57 votes, average: 4.57 out of 57 votes, average: 4.57 out of 5 (7 votes, average: 4.57 out of 5, rated)

Java date/time manipulation, formatting and conversion can be very tricky with things getting even more complicated when timezones are taken into consideration. A lot of developers (even seasoned ones) often run into problems when playing with date/time values.

Related Posts

[Environment Specific configuration for Java Applications](http://www.devrecipes.com/2009/08/14/environment-specific-configuration-for-java-applications/)  
[JSTL and Struts logic tags: a quick reference](http://www.devrecipes.com/2009/10/19/jstl-and-struts-logic-tags-iteration-quick-reference/)  
[Flex Chart from xml using Java Servlets](http://www.devrecipes.com/2009/05/10/flex-chart-from-xml-data-using-httpservice-with-java-servlets/)  
[Get System Information using Java](http://www.devrecipes.com/2009/07/19/get-system-information-using-java/)

Here’s a ready reference of some common date manipulation/formatting tricks that can be quite handy. They have been tested on **Jdk 1.5.**

* **Convert a String to a java.util.Date object:**

01.Date doStringToDate(String date,String inFormat){

02.DateFormat df = new SimpleDateFormat(inFormat);

03.Date dt=null;

04.try

05.{

06.dt= df.parse(date);

07.} catch (ParseException e)

08.{

09.//handle exception

10.}

11.return dt;

12.}

In the above, the inFormat parameter must exactly match the format of the date represented by the date parameter, for example if date is “13/12/2007″ then inFormat must be “dd/MM/yyyy”, if date is “13 Dec 07″ then format must be “dd MMM yyyy” and so on. See the javadocs for [SimpleDateFormat](http://java.sun.com/j2se/1.5.0/docs/api/java/text/SimpleDateFormat.html) for more.

* **Convert java.util.Date object to a String:**  
  Specify the format of the String that you want using the SimpleDateFormat.

1.String doDateToString(Date dt){

2.SimpleDateFormat formatter=new SimpleDateFormat("MMM d KK:mm a E Z");

3.String s=formatter.format(dt);

4.return s;

5.}

* **Convert a date String in one format to a different format:**  
  The following will convert the String inDate in inFormat to a date String in outFormat. Again remember that inFormat must exactly match the format of inDate.

01.String oneFormatToAnother(String inDate,String inFormat,String outFormat){

02.DateFormat inDf = new SimpleDateFormat(inFormat);//input date is in this format

03.Date dt=null;

04.String outDate;

05.try

06.{

07.dt= inDf.parse(inDate);

08.} catch (ParseException e)

09.{

10.//handle exception

11.}

12.DateFormat outDf=new SimpleDateFormat(outFormat);

13.outDate=outDf.format(dt);

14.return outDate;

15.

16.}

So calling the above method in the following way- oneFormatToAnother(”02/11/2009″,”dd/MM/yyyy”,”E MMM d KK:mm a Z”) will return “Nov 2 00:00 AM -0500″ if your JVM is in EST.

* **Date and Time calculation:**  
  Use the methods of the Calendar class if you need to perform date Calculations. An example is given below:

01.void dateCalculationExample(Date d1){

02.

03.Calendar c= Calendar.getInstance();

04.c.setTime(d1);

05.c.add(Calendar.YEAR, 2);

06.c.add(Calendar.HOUR,5);

07.c.add(Calendar.DAY\_OF\_MONTH,-4);

08.

09.Date d2=c.getTime();

10.System.out.println("modified date="+d2.toString());

11.

12.}

* **Timezone conversion of a date String:**  
  Changing the timezone of a Date object is not recommended and might produce unexpected results. What will work however is timezone conversion of a date string. The following will convert a date string ‘inDateStr’ which is in ‘inTz’ Timezone and in format ‘inFormat’ into another date string in ‘outTz’ timezone and formatted as in ‘outFormat’.

01.String changeTimeZone(String inTZ,String outTZ, String inDateStr, String inFormat,String outFormat){

02.//convert the incoming date string to a date object

03.Date inDate=null;

04.DateFormat inDf=new SimpleDateFormat(inFormat);

05.inDf.setTimeZone(TimeZone.getTimeZone(inTZ));

06.try{

07.inDate=inDf.parse(inDateStr);

08.}catch(ParseException e){

09.//handle exception

10.}

11.//System.out.println("inDate tostring="+inDate.toString());//Note this will give you time in default TZ and not in inTZ

12.

13.DateFormat outDf=new SimpleDateFormat(outFormat);

14.outDf.setTimeZone(TimeZone.getTimeZone(outTZ));

15.

16.String st=outDf.format(inDate);

17.return st

18.}

Example: changeTimeZone(”America/Denver”,”Asia/Calcutta”,”02/08/2010 15:00:20″,”MM/dd/yyyy HH:mm:ss”,”dd/MM/yy HH:mm:ss z”); will result in ‘09/02/10 03:30:20 IST’.

* **Convert timezones of a timestamp when storing into a database:**  
  Very often date/time information in an application must be stored in a fixed timezone even though users may enter time from any timezone. Therefore there is always a need to convert time into this fixed timezone. It is not a good idea to use the database default timezone in this case as this may change due to administrative reasons. Instead the conversion should be done explicitly.  
  If you have a date string ‘inDateStr’ in Timezone ‘inTz’ and need to convert it to a different timezone(outTz) when storing in a database column of type Timestamp use the following, paying particular attention to lines 24 and 25.

01.public void loadTimeInDB(String inDateStr, String inTz,String inFormat,String outTz){

02.

03.Connection oraConn=null;

04.PreparedStatement stmt=null;

05.Date inDate=null;

06.

07.//convert to Date object

08.DateFormat inDf=new SimpleDateFormat(inFormat);

09.inDf.setTimeZone(TimeZone.getTimeZone(inTz));

10.try{

11.inDate=inDf.parse(inDateStr);

12.}catch(ParseException e){

13.//handle exception

14.}

15.

16.try{

17.//do DB stuff

18.Class.forName("oracle.jdbc.OracleDriver");

19.oraConn = DriverManager.getConnection("jdbc:oracle:thin:@host:port:sid","username","pw");

20.stmt=oraConn.prepareStatement("insert into test\_date values(?,SYSDATE)");

21.

22.//timezone conversion takes place here with a Calendar Object

23.Timestamp dbTs=new Timestamp(inDate.getTime());

24.Calendar cal=Calendar.getInstance(TimeZone.getTimeZone(outTz));

25.stmt.setTimestamp(1,dbTs,cal);

26.

27.//update and close connection

28.stmt.executeUpdate();

29.stmt.close();

30.oraConn.close();

31.}catch(Exception e){

32.//handle exception

33.

34.}

35.

36.}

So if you call the above method with the parameters **loadTimeInDB(”02/08/2010 15:36:20″,”America/New\_York”,”MM/dd/yyyy HH:mm:ss”,”Europe/Paris”)** the timestamp column in the databse should show as **08-FEB-10 09.36.20.000000000 PM.** or something similar.  
**Caveat: The above was tested in Oracle 10g and did the conversion correctly, but in MySQL it did not. So use it carefully.** If this did not work for you, you might try manually calculating the offset of your target timezone as described next.

* **Convert timezones using offset and store in database:**  
  Timezone conversions can also be achieved by calculating the time difference between the two timezones (using java.util.Timezone’s getOffset() method) and adding this difference to the time that needs to be converted. So you might think this should be pretty straight forward.  
  *However in our example above, when you do a inDate.getTime(), this automatically converts the time given by ‘inDateStr’ and ‘inTz’ to the application’s (jvm) timezone*.  
  In other words if the time component ‘inDateStr’ is “16:15:20″ and ‘inTz’ is “America/Denver”, then when you do a inDate.getTime(), the time is implicitly converted to ‘18:15:20′ if your jvm’s default timezone is “America/New\_York” (+2 hours). **For this reason, the time difference between the target timezone ‘outTz’ and jvm’s timezone is used in the calculateOffset() method below, instead of the time difference between ‘inTz’ and ‘outTz’.** Note that this behavior might differ from jvm to jvm, so make sure to test thoroughly. The modified code is as below:

01.void loadTimeInDB(String inDateStr, String inTz,String inFormat,String outTz){

02.

03.Connection oraConn=null;

04.PreparedStatement stmt=null;

05.Date inDate=null;

06.

07.//convert to Date object

08.DateFormat inDf=new SimpleDateFormat(inFormat);

09.inDf.setTimeZone(TimeZone.getTimeZone(inTz));

10.try{

11.inDate=inDf.parse(inDateStr);

12.}catch(ParseException e){

13.//handle exception

14.}

15.

16.try{

17.//do DB stuff

18.Class.forName("oracle.jdbc.OracleDriver");

19.oraConn = DriverManager.getConnection("jdbc:oracle:thin:@host:port:sid","username","pw");

20.stmt=oraConn.prepareStatement("insert into test\_date values(?,SYSDATE)");

21.

22.//timezone conversion takes place here using the calculateOffset method

23.Timestamp dbTs=new Timestamp(inDate.getTime()+calculateOffset(inDate,outTz));

24.stmt.setTimestamp(1,dbTs);

25.

26.//update and close connection

27.stmt.executeUpdate();

28.stmt.close();

29.oraConn.close();

30.}catch(Exception e){

31.//handle exception

32.

33.}

34.

35.}

36.

37.long calculateOffset(Date onDate,String outTz){

38.//First get offset from UTC of Target Timezone

39.TimeZone Tz1=TimeZone.getTimeZone(outTz);

40.long outTzOffset= Tz1.getOffset(onDate.getTime());

41.

42.//Now get the offset from UTC of your application (JVMs) default timezone

43.Calendar cal=Calendar.getInstance();

44.TimeZone jvmTz=cal.getTimeZone();

45.long inTzOffset=jvmTz.getOffset(onDate.getTime());

46.

47.//calculate offset of target timezone from jvm timezone

48.long offset=(outTzOffset - inTzOffset);

49.//double offsetInHrs=offset/(1000\*60\*60); //offset in hours: use for debugging

50.return offset;

51.

52.}

So calling the above function with these parameters:**loadTimeInDB(”02/09/2010 22:34:20″,”Europe/Paris”,”MM/dd/yyyy HH:mm:ss”,”Asia/Shanghai”)** will store the time as **“10-FEB-10 05.34.20.000000000 AM”**in the database, which is the time that corresponds to 10:34 PM of the previous day (Feb 9th) in Paris.